Chapter 5

The Halting Problem

in general it is impossible to determine for which input strings p a Turing machine T will complete its computation and halt. The proof of this is reminiscent of the 'diagonal argument' used in the proof of Cantor's Theorem on cardinal numbers, which states that the number of elements of a set X is strictly less than the number of subsets of X.

5.1 Sometimes it *is* possible

It is easy to say when *some* Turing machines will halt. Consider for example our adding machine

$$T : \langle m \rangle \langle n \rangle \to \langle m + n \rangle.$$

We know that, for any Turing machine T, the input strings p for which $T(p)$ is defined form a prefix-free set.

But it is easy to see that the set

$$S = \{p = \langle m \rangle \langle n \rangle : m, n \in \mathbb{N}\}$$

is not only prefix-free, but is actually a maximal prefix-free set, that is, if any further string is added to S it will cease to be prefix-free. It follows that for our adding machine, $T(p)$ is defined precisely when $p \in S$.

Moreover, it is easy to construct a Turing machine H which 'recognises' S, ie such that

$$H(p) = \begin{cases} 1 \text{ if } p \in S, \\ 0 \text{ if } p \notin S. \end{cases}$$

This argument applies to any Turing machine

$$T : \langle n \rangle \to \langle f(n) \rangle,$$

where $f : \mathbb{N} \to \mathbb{N}$ is a computable function.

Perhaps surprisingly, there *are* Turing machines to which it does not apply.

5.2 The Halting Theorem

Proposition 5.1. *There does not exist a Turing machine H such that*

$$H(\langle T \rangle \langle p \rangle) = \begin{cases} 1 \text{ if } T(p) \text{ is defined} \\ 0 \text{ if } T(p) \text{ is undefined} \end{cases}$$

for every Turing machine T and every string p

Proof ▶. Suppose such a Turing machine H exists. (We might call it a 'universal halting machine'.)

Let us set $p = \langle T \rangle$. (In other words, we are going to feed T with its own code $\langle T \rangle$.) Then

$$H(\langle T \rangle \langle \langle T \rangle \rangle) = \begin{cases} 1 \text{ if } T(\langle T \rangle) \text{ is defined} \\ 0 \text{ if } T(\langle T \rangle) \text{ is undefined.} \end{cases}$$

Now let us modify H to construct a 'doubling machine' D such that

$$D(\langle T \rangle) = H(\langle T \rangle \langle \langle T \rangle \rangle)$$

for any machine T.

Thus D reads the input, expecting the code $\langle T \rangle$ for a Turing machine. It doesn't really matter what D does if the input is not of this form; we may suppose that it either tries to read past the end of the input, or else goes into an infinite loop. But if the input is of the form $\langle T \rangle$ then D doubles it, writing first $\langle T \rangle$ on the tape, followed by the string code $\langle \langle T \rangle \rangle$. Then it emulates H taking this as input.

Thus

$$D(\langle T \rangle) = H(\langle T \rangle \langle \langle T \rangle \rangle) = \begin{cases} 1 & \text{if } T(\langle T \rangle) \text{ is defined,} \\ 0 & \text{if } T(\langle T \rangle) \text{ is undefined.} \end{cases}$$

Finally, we modify D (at the output stage) to construct a machine X which outputs 0 if D outputs 0, but which goes into an infinite loop if D outputs 1. Thus

$$X(s) = \begin{cases} \perp & \text{if } D(s) = 1, \\ 0 & \text{if } D(s) = 0. \end{cases}$$

(We don't care what X does if $D(s)$ is not 0 or 1.) Then

$$X(\langle T \rangle) = \begin{cases} \perp & \text{if } T(\langle T \rangle) \neq \perp, \\ 0 & \text{if } T(\langle T \rangle) = \perp. \end{cases}$$

This holds for all Turing machines T, In particular it holds for the machine X itself:

$$X(\langle X \rangle) = \begin{cases} \perp & \text{if } X(\langle X \rangle) \neq \perp, \\ 0 & \text{if } X(\langle X \rangle) = \perp. \end{cases}$$

This leads to a contradiction, whether $X(\langle X \rangle)$ is defined or not. We conclude that there cannot exist a 'halting machine' H of this kind.

We improve the result above by showing that in general there does not exist a halting machine for a single Turing machine.

Theorem 5.1. *Suppose U is a universal machine. Then there does not exist a Turing machine H such that*

$$H(\langle p \rangle) = \begin{cases} 1 \text{ if } U(p) \text{ is defined} \\ 0 \text{ if } U(p) \text{ is undefined.} \end{cases}$$

Proof ▶. Substituting $\langle X \rangle p$ for p, we have

$$H(\langle \langle X \rangle p \rangle) = \begin{cases} 1 \text{ if } X(p) \text{ is defined} \\ 0 \text{ if } X(p) \text{ is undefined.} \end{cases}$$

Evidently we can construct a slight variant H' of H which starts by decoding $\langle \langle X \rangle p \rangle$, replacing it by $\langle X \rangle \langle p \rangle$, and then acts like H, so that

$$H'(\langle X \rangle \langle p \rangle) = \begin{cases} 1 \text{ if } X(p) \text{ is defined} \\ 0 \text{ if } X(p) \text{ is undefined.} \end{cases}$$

But we saw in the Proposition above that such a machine cannot exist.